Electronic Version
Stylesheet Version v1.1.1

Description

MODULAR DLL ARCHITECTURE FOR GENERATING MULTIPLE TIMINGS

BACKGROUND OF INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates generally to any DLL (Digital Locked Loop) architecture requiring the generation of multiple timings (timing signals), and more particularly pertains to a simple modular DLL architecture for an eDRAM (embedded Dynamic Random Access Memory) capable of producing any required number of eDRAM timings.

[0003] The present invention is applicable generally to any DLL (Digital Locked Loop) architecture requiring the generation of multiple timings, and is applicable to memory products in general, for example to provide timings for memory array cores and for input and output data, including embedded and non-embedded DRAMs and SRAMs (Static Random Access Memories) and memory controllers.

[0004] 2. Discussion of the Prior Art

[0005] DLL (Digital Locked Loop) architectures are frequently used to produce required timing signals; however, the use of several DLLs to accommodate multiple timing signals is expensive in terms of real estate on the chip.

[0006] The embedded DRAM design for one present state of the art ASIC (Application Specific Integrated Circuit) demands a simple architecture capable of producing any required number of embedded DRAM timings.

[0007] Figure 1 illustrates a typical prior art DLL implementation for use in a memory product, and Figure 2 illustrates timing waveforms illustrative of the operation of the prior art DLL implementation of Figure 1.

[0008] Referring to Figure 1, a variable delay line 1 is used to delay an edge of an incoming clock signal In by a desired amount. The variable delay line typically comprises either a variable number of fixed delay elements or a fixed number of variable delay elements. Typically a Mimic circuit 2 is used to determine how far ahead of the next clock edge an internal clock needs to transition. This is illustrated in Figure 2 by the designation Mimic Delay which shows the clock edge of the Result clock transitioning a given time ahead of the clock edge of the Out clock. It is common in

synchronous chip designs to require a synchronous circuit output be aligned or lined up with an external clock edge. A mimic circuit provides a fixed time delay which is a mimic of a parasitic circuit delay, and is used to determine when to drive the mimic circuit input so that the mimic circuit output and the synchronous circuit output are properly aligned with the external clock.

[0009] The output signal Out from the mimic circuit 2 should be delayed 360^{o} with respect to the main clock signal In, as illustrated in Figure 2 by the designation Total Delay = 1 cycle. The main clock input signal In and the output signal Out from the mimic circuit 2 are compared by a Phase Compare circuit 3, and the Phase compare circuit determines if the delay is too little or too much relative to the desired 360^{o} delay phase shift. The output of the phase compare circuit is directed to a Delay Control circuit 4 which increases or decreases the delay through the Variable Delay Line 1. When the phase of the system is locked, the output signal Result will be the desired Mimic circuit delay before the next clock cycle of In. This can then be used to time output data from a circuit, for example. It should be noted that the standard DLL implementation can only use one Mimic circuit, and only offers a single

output timing signal Result.

Summary of Invention

[0010] The present invention provides a single DLL architecture capable of generating any number of multiple timing signals.

[0011] The present invention provides a modular Digital Locked Loop (DLL) architecture capable of generating a plurality of multiple phase clock signals for synchronization of embedded DRAM systems with on chip timing. The architecture comprises a single core frequency locking circuit that includes a delay element with control logic and locking circuitry capable of locking the DLL system clock frequency to an external clock, and a plurality of secondary phase locking circuits capable of synchronizing a plurality of internal clock signals to any phase of the external clock.

Brief Description of Drawings

[0012] The foregoing objects and advantages of the present invention for a modular DLL architecture for generating multiple timings may be more readily understood by one skilled in the art with reference being had to the following detailed description of several embodiments thereof,

taken in conjunction with the accompanying drawings wherein like elements are designated by identical reference numerals throughout the several views, and in which:

[0013] Figure 1 illustrates a typical prior art DLL implementation for use in a memory product.

[0014] Figure 2 illustrates timing waveforms illustrative of the operation of the prior art DLL implementation of Figure 1.

[0015] Figure 3 illustrates an exemplary DLL architecture pursuant to the present invention that is capable of producing any number of required timing signals, and comprises a single core frequency-locking block and any number of secondary phase-locking blocks.

[0016] Figure 4 illustrates modular DLL architecture timing signals representative of the operation of the present invention.

Detailed Description

[0017] The present invention provides a modular DLL architecture that is capable of producing any number of required timing signals and comprises a single core frequency-locking block and any number of secondary phase-locking blocks.

[0018] One exemplary application of the present invention relates to the generation of timing signals on an embedded

DRAM, particularly embedded DRAM core array timings wherein a duty cycle of a clock includes an active portion or percentage of the total duty cycle and a restore portion or percentage of the total duty cycle, and the relative phase between the active portion and the restore portion must be precisely controlled.

[0019] Figure 3 illustrates an exemplary DLL architecture pursuant to the present invention that is capable of producing any number of required timing signals, and comprises a single core frequency-locking block and any number of secondary phase-locking blocks.

[0020] The single core frequency-locking block, designated FREQ Lock in Figure 3, has a system clock CLK input and an output designated CLK Phase Taps. The single core frequency-locking block can comprise a DLL circuit as shown in Figure 1 without the mimic circuit, such that the circuit comprises a DLL which includes a delay element, delay element controls, and locking circuitry which locks the DLL to the system clock frequency. The DLL of the core frequency-locking block then continuously updates the frequency lock as environmental conditions change, producing a very closely matched clock throughout the operation.

[0021] Traditional DLLs incorporate the mimic circuitry into the main DLL delay element, requiring multiple DLLs to form multiple mimic-path delays, whereas the main frequency-locking core block of the present invention does not incorporate the mimic circuitry therein, and so does not require multiple DLLs to form multiple mimic-path delays.

[0022] The frequency-locking core block FREQ Lock is used to produce a number of different clock phases that can be individually phase adjusted for any number of mimic-path delays.

[0023] The delay element itself of the frequency-locking core block FREQ Lock is designed to produce an output designated CLK Phase Taps which comprises a series of phase tap points that evenly divide the full 360 degree clock cycle into n degree steps, and in an example of n = 64, into 5.625 degree steps, although n can be any desired number. The delay element can comprise a fixed number of variable delay elements, and in one designed embodiment included a fixed number of variable delay elements providing 0 to 63 tap points, each separated by 5.625 degrees.

[0024] Figure 3 illustrates three exemplary secondary phase-locking blocks, an upper secondary phase-locking block

with a fixed time delay introduced by a Mimic circuit and producing a secondary output clock DOCLK, and two lower secondary phase-locking blocks without a fixed time delay introduced by a Mimic circuit and producing respective secondary output clocks PCLK and SDCLK, illustrating that the secondary phase-locking blocks can include a mimic circuit or not depending upon the particular application.

[0025] Any one of the 0 to 63 tap points can be chosen by each secondary phase locking block for a given clock edge to time a particular DRAM array timing (i.e. active/precharge, signal development, data output clock, etc.).

[0026] The most basic secondary phase locking block is shown as the middle secondary phase locking block which produces the clock output PCLK. The secondary phase locking block includes a block entitled Phase Lock which selects a particular one of the 0 to 63 taps and passes it to its output labeled Tap, which is directed to the D port of a latch L. The Phase Lock can comprise a MUX for selecting one of the 0 to 63 inputs and a sequencer and control logic for producing the count and controlling the MUX as described in the operation below. The latch L also receives the same input system clock CLK (as the input to the core frequency

locking block) at its clock port shown as >, and uses it along with the clock received at its D port, to produce an output to the DN (down) control of the Phase Lock.

[0027] In operation, the secondary phase lock block counts upwardly from 0 toward 63 in single steps. At each step, the system CLK rising edge at port > of the latch L causes the Latch L to sample the D port, and this operation continues until a low to high transition at the rising edge of the clock at the D port of the latch passes a low to high transition at the rising edge of the system CLK. The count is then reduced by 1, and the count then hovers about the count at which the two rising edges are closely coincident. For instance, suppose that the count starts at 0 and proceeds upwardly in increments of 1, and for each of counts 0 to 40, a sampling of the D port produced a low, and then at the count of 41, a sampling of the D port produced a high. The count would be reduced to 40, and the count would hover at 40 at the coincidence of the two rising edges. That count of 40 is a digital output of the Phase Lock labeled Offset which is the Tap Select input to a MUX (multiplexer) which selects and passes the 40 clock of the possible 0 to 63 clocks as the output at PCLK.

[0028] The lower secondary phase locking circuit producing the

clock SDCLK inserts, into the basic design explained above, an adder labeled + between the output Offset and the input Tap Select which simply adds a phase increment or offset of the system clock CLK to the clock output SD-CLK. For instance, continuing the above example where Offset is 40, a Phase Select Tap Input of a digital 10 would produce a Tap Select of 50, to cause the MUX to select and pass the 50 clock of the possible 0 to 63 clocks as the output at SDCLK.

[0029] The upper secondary phase locking circuit producing the clock DOCLK inserts a fixed time delay introduced by the Mimic circuit into the Phase Lock. The difference between the fixed time delay introduced by the Mimic circuit and the delay introduced by the adder labeled + is that the time delay introduced by the Mimic circuit is a fixed time delay, whereas the delay introduced by the adder labeled + is a relative time delay which is a phase increment or offset of the system clock CLK.

[0030] Any number of timing phases can be easily generated by either adding secondary phase locking circuits (if some mimic-path delays need to be changed), or by simply adding phase tap multiplexers, or multiplexers and adders, for generating multiple clocks from a single

mimic-path delay if that delay is common.

[0031] Figure 4 illustrates modular DLL architecture timing signals representative of operation of the present invention. Referring to Figure 4, note that the cycle is divided into 64 equal segments. Each "Tap" from the DLL delay line can be used as a potential phase for an internal clock timing. In Figure 4, two exemplary outputs are shown, Desired Tap1 and Desired Tap2. These two clocks will always transition high at Mimic Delay1 and Mimic Delay2, respectively, before the external clock.

[0032] In general, three key pieces of timing information are taken into account for each final DLL output.

[0033] The first key piece of timing information is the insertion delay of the frequency lock. Since the frequency lock is phase shifted, the first tap from the delay line is not aligned with the external clock. Therefore, to get an output that is in a known position relative to the external clock, the insertion delay of the frequency lock must be compensated. This is done in the core phase lock circuitry in Figure 3.

[0034] The second key piece of timing information is the mimic circuit delay which is optional. It is very common in synchronous chip designs to require a circuit output be lined

up with an external clock edge. The DLL can use a mimic of that circuit delay to determine when to drive it's input so the output is lined up with the external clock.

[0035] The third key piece of timing information is the control input to the DLL to provide the ability to reference each secondary phase lock block to any phase position of the external clock. In Figure 1, the prior art DLL system can only reference the mimic output to the rise of the external clock. The Modular DLL Architecture of the present invention allows referencing each secondary phase lock block, including a mimic output, to any phase of the external clock. This is accomplished using the Phase Tap Selects from Figure 3. An adder circuit + combines the desired tap point result (offset) from the phase lock circuitry with the Phase Tap Select input which represents the desired phase position as an external clock reference. This allows the Modular DLL Architecture to, for example, generate an internal data out clock that will cause external data transitions to occur at the external clock rise plus 90 degrees (change data 1/4 into the cycle). If the Phase Tap Selects are all zero, the DLL will use the external clock rise as the reference to "back up" from.

[0036] The present invention provides a simple, modular archi-

tecture wherein multiple timings are generated with a single DLL circuit. The unique design incorporates a single core frequency-locking block in conjunction with multiple phase lock/tap select units. Any number of phase lock circuits can be combined with mimic circuits to remove a variety of clock delays and produce a resultant series of delay adjusted clock phases.

[0037] The present invention provides a DLL architecture that provides for multiple output clock timings from a single synchronized system. The DLL architecture provides for multiple timings and comprises a single core frequency-locking block and any number of secondary phase-locking blocks. The core frequency-locking block includes a delay element, delay element controls, and locking circuitry to lock the DLL to the system clock frequency. The DLL architecture allows synchronization of clocks to any phase of the external clock.

[0038] While several embodiments and variations of the present invention are described in detail herein, it should be apparent that the disclosure and teachings of the present invention will suggest many alternative designs to those skilled in the art.